Problem 1:

Identify whether the given linked list is cyclic or not?

```
class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        slow=head
        fast=head
        while fast and fast.next:
            slow=slow.next
            fast=fast.next.next
            if slow==fast:
                return True
        else:
            return False
```

Problem 3:

Find the longest palindrome from the given string. Palindrome is a word, phrase, or sequence that reads the same backwards as forwards, e.g. madam, civic, radar

```
def printSubStr(str, low, high):


        for i in range(low, high + 1):
                print(str[i], end = "")
def longestPalSubstr(str):
        n = len(str)
```

```
        maxLength = 1
        start = 0
        for i in range(n):
                for j in range(i, n):
                        flag = 1

                        # Check palindrome
                        for k in range(0, ((j - i) // 2) + 1):
                                if (str[i + k] != str[j - k]):
                                        flag = 0

                        # Palindrome
                        if (flag != 0 and (j - i + 1) > maxLength):
                                start = i
                                maxLength = j - i + 1

        print("Longest palindrome subString is: ", end = "")
        printSubStr(str, start, start + maxLength - 1)

        # Return length of LPS
        return maxLength

# Driver Code
if __name__ == '__main__':
   str = "forgeeksskeegfor"
        print("\nLength is: ", longestPalSubstr(str))
```